Parallel image classification on theHIVE

Mike Smit[1], *Jerry Garegnani*[2], *Matt Bechdol*[1] *and Samir Chettri*[1]

[1] Global Science & Technology at Code 935 NASA/GSFC, 6411 Ivy Lane, Greenbelt, MD 20770.

[2] NASA/GSFC, Applied Information Science Branch, Code 935, Greenbelt, MD 20770.

ABSTRACT

Remotely sensed imagery represents a growing source of information to many practical applications. Technologies to rapidly process imagery data into useful information products has not kept pace with the rapidly growing volume and complexity of imagery data increasingly available from Government and commercial sources. Significant processing speed improvements have been achieved by implementation of classification methods on the Highly-parallel Integrated Virtual Environment (HIVE) - a Beowulf class system using Parallel Virtual Machine (PVM) software. This paper discusses our parallel processing architecture and how three different classification algorithms performed in this computing environment. Also discussed are conclusions and recommendations for future work to apply these techniques to more complex data an further improve the processing speeds.

Keywords. theHIVE, classification, parallel algorithms, Neural Networks, Gaussian Maximum Likelihood Classifier, Polynomial Discriminant Method, Mixture Model Neural Network, MIMD, SPMD.

1. Introduction

Data volume has been growing exponentially due to increasingly sophisticated methods of data collection. Whether it be from medical imagery or satellite instrumentation; the amount of data being generated has begun to tax the capacity of modern processing and storage equipment. The increased spatial and spectral resolution of emerging satellite sensors significantly increases the volume of information that can be generated from the data. The value of the information derived from imagery increases significantly with increased timeliness of delivery.

As more sources of imagery become available, there will not only be a data deluge - creating a massive problem of storage and retrieval, but also processing bottlenecks since information needs to be generated very rapidly. Though the problem of storage and retrieval are daunting issues, we will not deal with these directly. Rather, we will examine methods of data analysis that improve speed for more timely delivery of information from raw imagery data. Thus in this paper we will consider the classification of remotely sensed images - in particular the parallelization of three methods that increases processing speeds.

The following sections of the paper describe:

1. The basics of parallel processing and in particular the paradigm of single program multiple data (SPMD) used in this paper.
2. The pattern recognition problem as it pertains to image classification and contains a brief discussion of the three classification methods used in our computational experiments. The mathematics and accuracy analysis of these three methods (Gaussian Maximum Likelihood Classification, Polynomial Discriminant Method and the Mixture Model Neural Network) are discussed in greater detail in [4].
3. How the parallel code was applied to classify a large LANDSAT scene that we have in a data storage and retrieval system developed at Code 935 called RODIN [16].
4. The issues that have been raised by the present work.

2. Parallel processing background

Parallel processing has been defined in [2] as: *A large collection of processing elements that can communicate and cooperate to solve large problems fast.* This definition makes no specification of the processing elements, their organization, the communication channels or the organization and manner in which these may cooperate.

Conceptually, parallel processing may be thus seen to consist of the following steps:

1. Partition a task into smaller tasks.
2. Send subtasks to multiple processors.
3. Each processor (running a subtask) receives needed data.
4. Each machine independently processes its data.
5. If necessary exchange data through communication channel (network).
6. Continue computations.
7. Integrate results from processors and present them to user.

In [1] a taxonomy of parallel architecures is described. This is a macroscopic classification in which the name given to an architecture depends on how the instructions are related to the data being processed. Figure 1 below shows the relationships of the various types of architectures.

	Number of Data Streams	
	Single	Multiple
Number of Instruction Streams	SISD	SIMD
	MISD	MIMD

Figure 1: Flynn's taxonomy. *SISD* = Single Instruction Single Data, *SIMD* = Single Instruction Multiple Data, *MISD* Multiple Instruction Single Data, *MIMD* Multiple Instruction Multiple Data.

1. **SISD – Single Instruction Single Data**. Conventional computers (i.e., the workstation on your desk) have one stream of instructions. Each instruction works on a single data stream.
2. **SIMD – Single Instruction Multiple Data**. Vector processing machines are SIMD machines. Here each element of a vector (we consider scalars to be degenerate vectors) is broken up into separate streams. However, a single instruction stream remains, i.e., multiplication of each element of a vector by constant.
3. **MISD – Multiple Instruction Single Data**. This item is generally considered void. See [3] for details.
4. **MIMD – Multiple Instruction Multiple Data**. This item consists of groups of linked processors (recently linking has been done by high speed networks). Each processor has its own instruction stream.

Both SIMD and MIMD have come into common usage in computer science. Below we describe a second level taxonony for MIMD [14].

1. **DM–MIMD - Distributed Memory MIMD**. In this variant of MIMD, individual processors have their own memory and CPU shown in Figure 2. Every computer can talk to every other computer via the network.
2. **SM–MIMD - Shared Memory MIMD**. Figure 3 shows how each processor can access a common memory via a high speed network. Such a configuration is frequently called a *symmetric multiprocessor*.

2.1. Message passing and SPMD

Recently, DM–MIMD has become one of the most popular parallel proceeing architectures. As explained earlier, this consits of independent processors each with its own memory

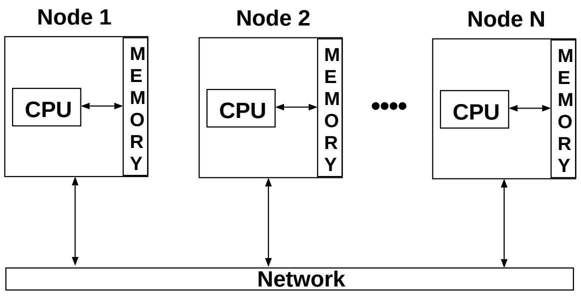

Figure 2: Block diagram of DM–MIMD.

and a copy of its own program as well as data. Messages (for data exchange and control) are sent between processors (senders and receivers). The system we have described is called a *message–passing system*. A particular form of message–passing systems is called **SPMD** – Single Program Multiple Data [1]. What this acronym means is that the same program is running on multiple machines (nodes) and data is given to each node for processing. Note that the data itself may be passed around between nodes.

A particular variant of SPMD is the *master–slave* paradigm. Here the master is in charge of talking to and distributing the work to each slave. Usually, the slaves communicate with only the master. A little thought shows that point based image processing (i.e., where each pixel in an image is treated by the same method) is easily amenable to master–slave based parallel processing. Here the image would be broken up into sections, passed to the various nodes each of which would perform the same operations on the sub–sections. The results would be passed back to the master for reassembly. In the extreme case each pixel could be passed to an individual processor. [2]

2.2. theHIVE

theHIVE (Highly-parallel Integrated Virtual Environment) is a low cost, coarse grained parallel processing machine using

[1]This is a subset of of the DM–MIMD style computing

[2]There are natural limitations to this, noted in subsequent sections.

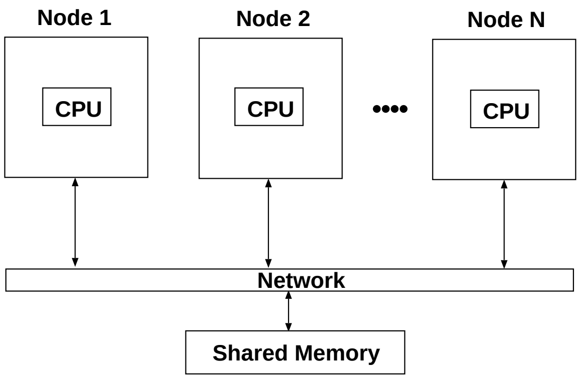

Figure 3: Block diagram of SM–MIMD.

Item	Qty	Description
Processor	2	Dual Pentium 2, 200 MHz
RAM		64 MB
Hard Disk		2.5 Gigabyte/node
Network		100 Base T Ethernet card
OS		LINUX

Table 1: DRONE description

off the shelf hardware to keep prices down. Currently the configuration consists of 64 dual processor Pentiums (refered to by its developers as BEES - Busy Environmental Entity) running LINUX connected by a 100baseT private ethernet LAN. Additionally, there are two types of front–end nodes that researchers use to program the BEES. The first type is known as QUEEN (QUasi Enabler of ENtities) and is used to administer the entire HIVE while the DRONEs (DRiver Of Nominal Entities) are portals into theHIVE and the BEES. theHIVE has either the Parallel Virtual Machine (PVM) or Message Passing Interface (MPI) software thereby enabling parallel programming using a message passing paradigm. It runs on the REDHAT version of LINUX. Additionally there are one or more queens and drones. The queen administers theHIVE and the drone permits access to it. The configuration of BEES and DRONES is shown in Table 1.

3. The pattern recognition problem

Figure 4 shows the standard pattern recognition box diagram where P stands for pattern space, F for feature space and C for classification space. This section addresses issues dealing with transferring a signal from the pattern space through feature space and on to classification space. In each transfer, it is hoped that there will be an increase in intelligibility of the signal. Typically as we go from P to F there is a reduction in dimension of the original signal and from F to C we are actally placing the signal in one or more pre–defined classes that make sense to the user of the system. Naturally, we may wish to go directly from P to F. Nothing in our model precludes this.

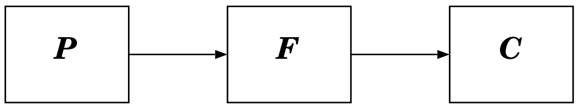

Figure 4: The pattern recognition problem

For example, one way to proceed from P to F, i.e., from pattern space to feature space is via principal components analysis. Dimensionality reduction is achieved via projection of the original data onto the principal subspaces. Usually, the first few principal components contain most of the information contained in the original bands. A good reference for this and other transformations from P to F is [8].

Similarly we can go from F (feature space) to classification space C. This requires a general pattern recognition algorithm. Discriminant functions are a particularly convenient way to proceed. Figure 5 schematically shows how the transformation from F to C is achieved by using, for example, a Neural Network ($\mathrm{M}^2\mathrm{N}^2$) [4].

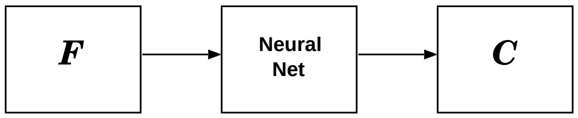

Figure 5: Going from F to C

3.1. The classifiers

Discriminant functions. In succeeding paragraphs we will discuss image classifiers based on pattern recognition techniques that use statistical discriminant functions. The job of designing the pattern classifier consists of first dividing the feature space into regions and then constructing the classifier so that it will identify any measurement vector as belonging to the class corresponding to the region in which it falls. These regions are sometimes also known as decision regions. Discriminant functions are part and parcel of classifier design. Much of the discussion in this subsection comes from several sources i.e., [5], [6], [7] and from a tutorial, survey and some very recent results in [9].

Assume there are K classes and that the K decision regions corresponding to these classes have been determined [3] Suppose we can find a set of K functions of $\mathbf{X}$, which we denote by $g_1(\mathbf{X}),\ g_2(\mathbf{X}),\ \cdots\ g_K(\mathbf{X})$, having the property that $g_k(\mathbf{X})$ has a larger value than any other of these functions whenever $\mathbf{X}$ is a point in the k^{th} decision region. These functions are referred to as discriminant functions. Then, if we wish to classify *any* $\mathbf{X}_u$, that is, determine to which decision region $\mathbf{X}_u$ belongs, all we need to do is calculate the values of $g_1(\mathbf{X}_u),\ g_2(\mathbf{X}_u),\ \cdots\ g_K(\mathbf{X}_u)$. The point $\mathbf{X}_u$ belongs to the class having the largest g value. Formally we can write:

$$\textit{Let } S_k \textit{ denote the } k^{th} \textit{ class. Decide } \mathbf{X} \in S_k \textit{ iff} \\ g_k(\mathbf{X}) \geq g_j(\mathbf{X}) \textit{ for all } j = 1, 2, \cdots K.$$

Two discriminant functions can have equal values only at the borders between their decision regions. A tie breaking rule must be defined for these cases.

In order to extend the discussion to statistical discriminant functions let $f(\mathbf{X} \mid S_k)$ be the probability density function (pdf) associated with the measurement vector $\mathbf{X}$, given that $\mathbf{X}$ is from class k. Let $P(S_k)$ be the *a priori* probability of class k. We can use the **maximum likelihood decision rule** [4] to identify the class to which $\mathbf{X}$ belongs. It can be stated as follows:

$$\textit{Decide } X \in S_k \textit{ iff} \\ f(\mathbf{X} \mid S_k)P(S_k) \geq f(\mathbf{X} \mid S_j)P(S_j), j = 1, 2, \cdots K \ .$$

[3]This is known as training. For each of the three classifiers we have implemented we will discuss the relevalt training steps.

[4]Sometimes known as the *maximum a posteriori* (MAP) rule.

In case we have no *a priori* knowledge the usual rule is to assume a uniform probability distribution, i.e., equally likely prior probabilities.

Below we discuss three classifiers that we have used for our experiments. In each case the mathematical formulation for training and classification is shown.

The gaussian maximum likelihood classifier (GMLC). This classifier assumes that data from any given class can best be described by a multivariate Gaussian, i.e., $\mathcal{N}(\mathbf{U}, \mathbf{\Sigma})$.

Classification. The discriminant function in this case is:

$$f(\mathbf{X}|S_k) = \ln[P(S_k)] - \frac{1}{2}\ln|\mathbf{\Sigma}_k| - \frac{1}{2}(\mathbf{X}-\mathbf{U}_k)^T\ \mathbf{\Sigma}_k^{-1}\ (\mathbf{X}-\mathbf{U}_k). \tag{1}$$

Where, $\mathbf{U}_k$ is the sample mean vector and $\mathbf{\Sigma}_k$ is the sample variance–covariance matrix of the kth class.

Training. Of course this begs the question: How are the estimates of $\mathbf{\Sigma}_k$ and $\mathbf{U}_k$ generated? They are generated from the exemplars $\mathbf{u}_m^{(k)}$, where $\mathbf{u}_m^{(k)} = [u_{1m}^k\ \cdots\ u_{rm}^k\ \cdots\ u_{Rm}^k]^T$. Here, k indexes the class and m indicates the m^{th} prototype of class S_k. We also have a count of the total number of exemplars from class S_k, denoted by M_k. The formulae for the mean and covariance vector are:

$$\mathbf{U}_k = \mathcal{E}\{\mathbf{u}\} = \frac{1}{M_k}\sum_{m=1}^{M_k}\mathbf{u}_m^{(k)} \tag{2}$$

$$\mathbf{\Sigma}_k = \mathcal{E}\{(\mathbf{u}-\mathbf{U}_k)(\mathbf{u}-\mathbf{U}_k)^T\} = P\sum_{m=1}^{M_k}(\mathbf{u}_m^{(k)}-\mathbf{U}_k)(\mathbf{u}_m^{(k)}-\mathbf{U}_k)^T. \tag{3}$$

Here, $P = \frac{1}{M_k - 1}$.

The polynomial discriminant function (PDM) The PDM represents a polynomial approximation to the probabilistic neural network (PNN). The PNN was found to be exorbitantly slow in the classification phase (though rapid in the training phase). Details of the PNN can be found in [12] and [13].

Classification. The PDM is written in the form of a polynomial upto terms of order r, the k^{th} discriminant function can be written in polynomial form as:

$$\begin{aligned} f(\mathbf{X}|S_k) &= D_{0\ldots0}^k + D_{100\ldots0}^k x_0 + \ldots + D_{000\ldots1}^k x_{d-1} \\ &\quad + \ldots + D_{z_0 z_1 \ldots z_{d-1}}^k x_0^{z_0} x_1^{z_1} \ldots x_{d-1}^{z_{d-1}} + \ldots \end{aligned} \tag{4}$$

As a consequence of our constraint on the polynomial (4), the constraints on z_l (in any of the terms in equation (4)) are given by $\sum_{l=0}^{d-1} z_l \le r$.

Training. The general term in (4) is given by

$$\begin{aligned} D_{z_0 z_1 \ldots z_{d-1}}^k &= \frac{1}{P_k}\frac{\sigma^{-2r}}{z_0! z_0! \ldots z_{d-1}!} \times \\ &\quad \sum_{m=1}^{P_k} W_{km0}^{z_0} W_{km1}^{z_1} \ldots W_{km(d-1)}^{z_{d-1}} \exp\left[C_{km}\right] \end{aligned} \tag{5}$$

$\mathbf{W}_{ki}$ is the i^{th} training pattern from the $0 \le k^{th} \le M-1$ category, P_k is the total number of training patterns in class k. Also, $C_{ki} = \mathbf{W}_{ki}^T\mathbf{W}_{ki}/2\sigma^2$ and σ is a parameter that is prechosen. Note the difference between $\mathbf{W}_{ki}$ which is a vector an W_{km0} which is a scalar and represents the first (zeroth) band of the mth sample from the kth class. For a detailed discussion of the PDM see [9].

The mixture model neural network ($\mathbf{M^2N^2}$) The mixture model neural network is a step toward reducing the complexity of the PNN while eliminating the problems associated with the PDM. These issues are discussed in detail in [4] and [9].

Classification. The mixture distribution (M^{2}N^2) is written as:

$$f(\mathbf{X}|S_k) = \sum_{j=1}^{M} f(\mathbf{X}|j, S_k)\alpha_j \tag{6}$$

Where, $\mathbf{X}$ is the input vector, α_j is the weight of the j^{th} density and S_k is the k^{th} class. As mentioned earlier we shall assume that all the components of the density are Gaussian having mean vectors $\mathbf{u}_j$ and variances $\mathbf{\Sigma}_j = \sigma_j\mathbf{I}$, where $\mathbf{I}$ is the identity matrix, i.e., assuming a circularly symmetric Gaussian. This is written as:

$$f(\mathbf{X}|j, S_k) = (2\pi\sigma_j^2)^{d/2}\exp\left\{-\frac{(\mathbf{X}-\mathbf{u}_j)^T(\mathbf{X}-\mathbf{u}_j)}{2\sigma_j^2}\right\}. \tag{7}$$

Training. The fundamental issue with the M^{2}N^2, equation (6), is determining the parameters $\mathbf{\Theta} = \{\mathbf{u}_j, \mathbf{\Sigma}_j, \alpha_j\}$ of the mixture distribution. In neural network terminology these would be the "weights." These are determined by the Expectation Maximization (EM) algorithm that has found use in a variety of fields including signal and image processing in particular, and in general in any field where maximum likelihood estimates have to be calculated from incomplete data. Good references for the EM algorithm are [10], [11] and [9].

4. Application

The classification methods discussed in the previous section were parallelized for both training and classification on the-HIVE. For training, each class statistic, for example the mean vector and variance and covariance matrix of the gaussian maximum likelihood classifier (GMLC) were obtained by having a single machine handle all the training data for that class. Thus, if there are N classes, N machines were used.

For classification, Figure 6 shows our processing steps. we used a data archiving, querying and retrieval system called RODIN (Regional Observation Digital Information Network) [16] to obtain our data which consisted of a LANDSAT scene. Once the data was available on the master on theHIVE it was broken up into smaller pieces and each piece sent to a node. Classification proceeded independently on each machine with the results being sent back to the master for eventual delivery to RODIN.

Details of RODIN, the data sets from RODIN and performance results are given in the sections that follow.

4.1. Regional Observational Digital Information Network - RODIN

A brief explanation of the RODIN [16] system is in order since as it was used in our computational experiments. In fact, as we shall see, our use of RODIN will illustrate certain important performance issues.

RODIN is a generic configurable data system for:

1. Capturing near–real–time spatially indexable data from a variety of sources (i.e., GIS, image, and point data sets are all handled by RODIN).
2. Automatically processing the data into useful products.
3. Indexing the metadata.
4. Storing the data and metadata.
5. Processing client queries against the metadata.
6. Retrieving and delivering the data and metadata to clients.

RODIN is:

1. Data format neutral - it can be configured to handle any format with data stored in files.
2. Built on open source tools, so it can be installed for very low cost.
3. Built with an open architecture, therefore custom clients can be built to do specific processing on the data.

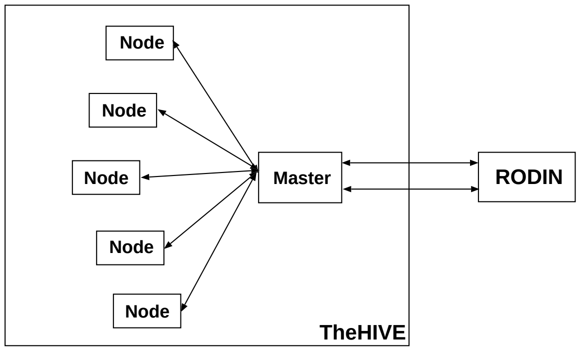

Figure 6: How theHIVE was used in our processing.

4.2. Test data

Initial test data for this work consisted of a Landsat 7 scene Path 33 Row 33, collected on July 28, 1999. The scene covers 34,225 square km of the Baltimore–Washington region during a substantial drought. Landsat 7 data is comprised of the band structure seen in Table 2. Due to differences in ground resolution, spectral overlap, and desire to keep processing time manageable, only bands 1 through 4 were implemented in the testing process. The test data cube was thus comprised of 4 × 8091 × 7301 (bands × columns × rows). An image of this region is shown in Figure 7.

In the near future the authors will be working with other types of remotely sensed data to highlight other data volume

Band	Spectral Range μmeters	Ground Resolution (m)
1	0.45 – 0.515	30
2	0.525 – 0.605	30
3	0.63 – 0.690	30
4	0.75 – 0.90	30
5	1.55 – 1.57	30
6	10.4 – 12.5	60
Pan	0.52 – 0.90	15

Table 2: Landsat bands

issues dictated by sensor characteristics. These datasets will represent high-resolution imagery (4 meter ground resolution IKONOS from Space Imaging, Inc.) and high-dimensional hyperspectral data (1, 2, and 3 meter ground resolution, upto 40 band data AISA: Airborne Imaging Spectrometer for Applications flown by 3DI of Easton, Maryland).

Selection of training data. The scene consists of many landcover/landuse types; the megalopolis of the Washington DC and Baltimore is obvious in the scene as is the highly recognizable Chesapeake Bay. Other significant features are the predominantly agricultural areas of Maryland's Eastern Shore and the eastern ridges of the Appalachian in the western part of the scene. The final major class in the scene in the large cloud area in the northwest portion of the Landsat image. Cloud areas were not masked out prior to classification in order to test the functionality of the parallel classifications first. These areas will later be masked to better assess the accuracy of methods discussed herein.

To provide sufficient training points for the classifiers, it was determined that 50×N points, where N equals the number of

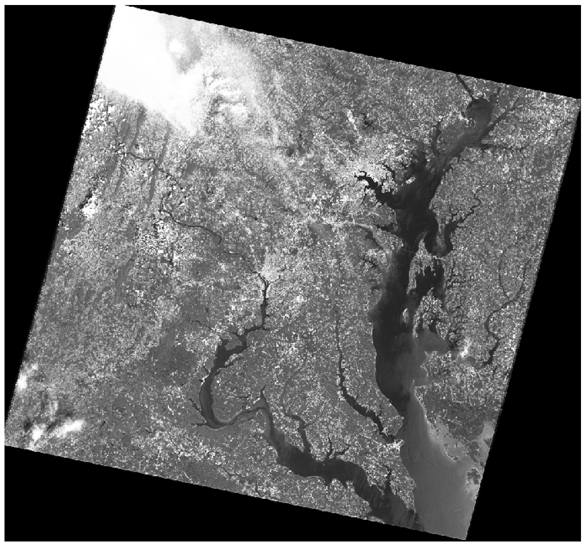

Figure 7: Landsat scene of the Baltimore–Washington area from July 28, 1999. This is a gray level version of a three band color composite.

dimensions or bands would adequately describe the scene using the Anderson Level 1 classification scheme [15] [5]. In this scene only the following classes were observed: Urban/Built Up, Agriculture, Forest, Water, Clouds, and Fallow. We designated 200 points for each class as training pixels. Due to the drought, there is potential for increased confusion among classes, as dry grasses, bare soil, and many urban environments are spectrally similar during these conditions.

4.3. Performance results

Previous subsections described the parallel classifiers used, the data system used (RODIN) as well as the data sets used in the experiment. In this sections timing results are presented. There is no discussion of the accuracy of the classifiers as that is not within the scope of this paper.

Experiments were carried out in the following manner: The parallel code for the classifiers (GMLC, M^2N^2 and PDM) was run on 4, 8, 16, 32 and 64 nodes of theHIVE. Each experiment was repeated ten times in order to get averages and standard deviations (representing in some sense the variation around the means). Figures 8 and 9 show the kinds of speedups we might expect along with error bars. Note, the speedups indicated in Figure 9 are the ratios of the time taken for the algorithm on four machines to the time taken on sixty–four. The reason for using 4 machines is because the limited disk–space and main memory of a single node on theHIVE prevented us from using one, two and three machines. Naturally, if one machine was used the speedups would be higher.

Figure 8 shows the *total* time taken for processing. The actual processing involved [6]:

1. Requesting data from RODIN in chunks - since the whole multi–band image will not fit into memory on any one node on theHIVE.
2. Sending chunks to nodes for processing with the clasification algorithm (GMLC, M^2N^2, PDM).
3. Returning classified chunks to the master for assembly.
4. The entire (one–band) classified image is sent to RODIN.

Figure 9 shows the *total* time taken for processing *minus* the time spent communicating with RODIN. There are two essential differences between Figures 8 and 9. Firstly there are the higher execution times. This is a natural consequence of the method that was used to calculate the values that make up the respective graphs. The second difference is more illuminating - namely there is very little variance in Figure 9 but a large variance in Figure 8 (around the respective means). This is a result of the communication uncertainty in the network between RODIN and theHIVE. theHIVE has an internal 100baseT network. This is also true of the network between RODIN and theHIVE. However, this link has much more traffic than theHIVE's internal network and thus there is greater variablity in execution times.

[5]The Anderson Level 1 scheme defines the following classes - urban, agricultural land, rangeland, forest land, water, wetland, barren, tundra and perennial snow or ice.

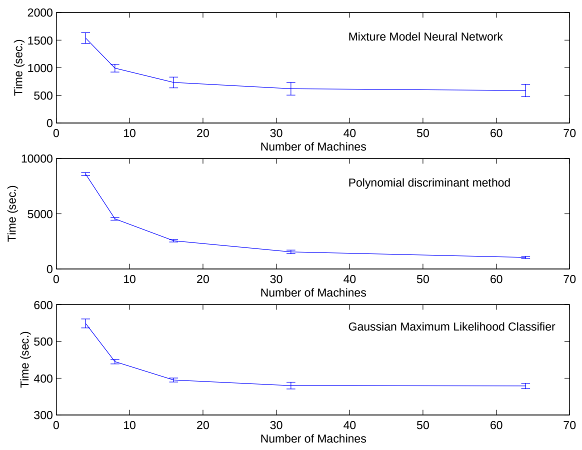

Figure 8: Speedups on theHIVE for several classifiers. This set of graphs shows the *total* time to process the data. See text for details.

5. Conclusions and future work

As data rates, number of bands and image sizes continue to increase, methods to process the data must also keep pace. The method for parallel image classification of very large data sets, described in this paper, is one such way to meet the above mentioned goal.

There are several issues raised by this research that points to further interesting research work. These are:

1. Figures 8, 9 show a stage of diminishing returns. At this point, the image chunk size has reached a small enough size that data transmission takes up a significant portion of the time while processing has reached

[6]Also see Figure 6 for details

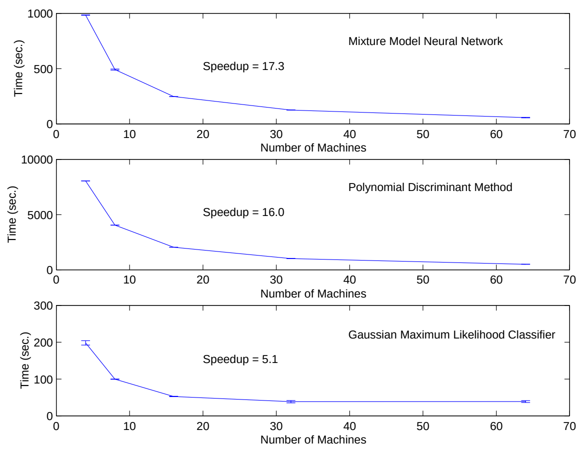

Figure 9: Speedups on theHIVE for several neural networks. This set of graphs shows the time to process the data *minus* the time spent communicating with RODIN. See text for details.

a minimum. Naturally, using a faster internal network would prevent the curves from flattening out between 16 and 32 machines. Myrinet [17], [18] is one such network that is becoming popular on Beowulf class machines.

2. The previous design could be sped up considerably by a slight reworking of the architecture. Figure 10 shows the alternate configuration. The key difference between this and the one in Figure 6 is that the master has been broken up into two entities. The first one (*distributing master*) upon being requested by an external client, requests data from RODIN, and sends it to the nodes for processing. At the same time it tells the *receiving master* to expect results from the nodes. The receiving master accepts results from nodes and either assembles the final classified image for sending to RODIN or the client, or it sends it piece–by–piece to a website for viewing.

3. The methods described in this paper are only suitable for classification of multi–spectral data from say, Landsat or IKONOS. Different algorithms will have to be adopted for hyperspectral data. GST at Code 935 has already begun work [19] in this area. At present we are contemplating parallelizing the Support Vector Machine [19] and the Maximum Entropy Method [20] for mixed pixel analysis. These algorithms have been applied to AISA (upto 40 bands, flown by a commercial company) and AVIRIS (200 plus bands, flown out of JPL) sensors.

4. An area where great progress may be made is in the area of parallel classification of the same image using multiple classifiers (such as the GMLC, PDM and M^2N^2). The results may be combined to produce a better classification. Additionally, if the same image has different (non–sequential) algorithms applied to it, the SPMD (single program multiple data) model may be applied. Finally, training and testing of classifiers is an important issue. One of the more important techniques is the leave–one–out method [21]. The principle is very simple and involves taking $n-1$ points from the sample and training the classifier on that information. The n^{th} point is classified and this training and testing procedure is repeated for all n points in the set. Quite clearly parallelism will reduce the waiting time for accuracy estimates in such a case.

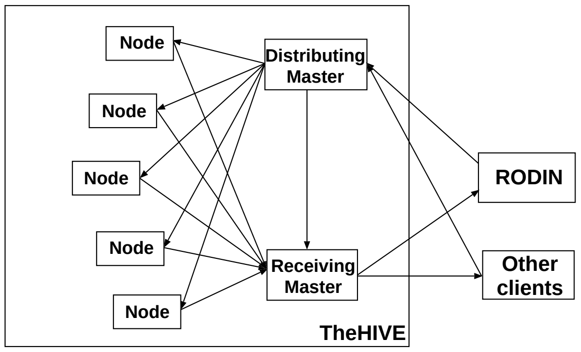

Figure 10: A new architecture to speedup processing. See text for details.

6. Acknowledgements

The authors would like to thank Jeff Love, Bob Bane, Claudia Castaneda and Idriss Mekrez for their work on RODIN and clients external to RODIN. Many thanks are due to GST's Director of technical programs, Dr. Celeste Jarvis, for keeping it all together.

References

1. Flynn, M. "Some computer organizations and their effectiveness." *IEEE Trans. Comput.*, Vol. C–21, 1972, pp. 948–960.
2. Almasi, G. S. and Gottlieb, A. *Highly Parallel Computing*, Benjamin/Cummings, Redwood City, California, 1989.
3. Hockney, R. W. and Jesshope, C. R. *Parallel Computers: Architecture, Programming and Algorithms*, Adam Hilger Ltd., 1981.
4. Chettri, Samir, Murakami, Y., Nagano, I. and Garegnani, J. "Comparing the computational complexity of the PNN, the PDM and the MMNN (M^2N^2)." In Selander, J. M. editor, *26th AIPR Workshop - Exploiting New Image Sources and Sensors.*, Volume 3240, 1997, pp. 126–132.
5. Meisel, W. S. *Computer–Oriented Approaches To Pattern Recognition*, Academic Press, 1972.
6. Andrews, H. C. *Introduction To Mathematical Techniques In Pattern Recognition*, Wiley–Interscience, 1972.
7. Richards, J. A. and Jia, X. *Remote Sensing Image Processing*, Wiley–Interscience, 1999.
8. Mather, P. M. *Computer processing of Remotely–Sensed Images.* John Wiley and Sons, Chichester, 1999.
9. Chettri, S. R. and Gualtieri, J. A. "Issues in the classification of remotely sensed images." Working paper, Applied Information Sciences Branch, Code 935, NASA/GSFC, Greenbelt, MD 20770, 2000.
10. Meng, X. L. and Pedlow, S. "EM: A bibliographic review with missing articles." *Proc. Statist. Comput. Sect. Am. Statist. Ass.*, Vol. 39, 1992, pp. 24–27.
11. Dempster, A. P. and Laird, N. M. and Rubin, D. B. "Maximum Likelihood from Incomplete Data via the EM Algorithm (with discussion)." *J. Roy. Statist. Soc. Series B*, Vol. 39, 1977, pp. 1–38.
12. Specht, D. "Probabilistic Neural Networks." *Neural Networks*, Vol. 3, 1999, pp. 109–118.
13. Chettri, S. R. and Cromp, R. F. "Probabilistic neural network architecture for high–speed classification of remotely sensed imagery." *Telematics and Informatics*, Vol. 10, Number 3, 1993, pp. 187–198.
14. Morse, H. S. *Practical Parallel Computing*, AP Professional, 1994.
15. Anderson, J. R. et al. "A land use and land cover classification system for use with remote sensor data." *Geological survey professional paper 964*, U.S. Government Printing Office, Washington, DC, 1976.
16. Bane, J. R., Love, J., Campbell, W., Garegnani, J., et al. "Regional Observation Digital Information Network (RODIN) 2000." Working paper, Applied Information Sciences Branch, Code 935, NASA/GSFC, Greenbelt, MD 20770, 2000.

17. Sterling, T. L. et al. *How to Build a Beowulf. A guide to the implementation and application of PC clusters.* The MIT Press, Cambridge Massachusetts, 1999.

18. VITA Standards Organization "Myrinet–on–VME: Protocol specification draft standard." Draft 1.1, 31 August 1998. Available at *http://www.myri.com/open-specs/myri-vme-d11.pdf*

19. Gualtieri, J. A., Chettri, S. R. and Cromp, R. F. et al. "Support Vector Machines Classifiers as applied to AVIRIS data." In *The Proceeding of the 1999 Airborne Geoscience Workshop*, JPL, February 8–11, 1999.

20. Chettri, S. R., Garegnani, J., Robinson, J., Coronado, P., Cromp, R. F., Netanyahu, N., Cambpell, W. J. "Multi–resolution maximum entropy spectral unmixing." In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 1997, pp. 347–352.

21. Weiss, S. M. and Kapouleas, I. "An empirical comparison of pattern recognition, neural nets, and machine learning classification methods." In *Eleventh Int. Joint Conference on Artificial Intelligence.*, 1989, pp. 781–787.